EDG/ECHO Integration Design Briefing

The EDG Team
GST, Inc.

Primary question: How do we get the EDG to talk to ECHO as quickly as possible?

- Problems to be addressed
  - ECHO has no valids
  - ECHO speaks XML, not ODL
    - Queries have to got out as AQL wrapped in XML
    - Results come back in XML

Basic ECHO access through the EDG

- ModeMINIECHO
  - Really basic interface - load or type in a query, send it to ECHO, display the XML returned
  - Necessary to validate:
    - ECHO SOAP interface
    - Availability/dependability of ECHO servers
    - Baseline for testing queries and seeing the results
    - Useful for checking up on ECHO

Getting ECHO-compatible valids

- ECHO states they will not provide valids
- Out of date anyway
- So we have to make our own

ModeMAKEVALIDS

- Ed discovered that using '%' (wildcard) as the dataset name in an ECHO AQL query gets all of the datasets
- ModeMAKEVALIDS
  - Adaptation of ModeMINIECHO
  - Loads one of these searches and sends it
  - Parses the returned XML with XML::Parser(Style=>Objects), creating a tree of Perl objects
  - Does a recursive treewalk to extract data from the tree, inserting it into a new IU::ODL object
  - Saves this as a new set of valids

Getting to a search

- Given valids, we can go from there to the creation of a query before hiting another problem
- The EDG would normally transform this to ODL and send it out via imsclienti (which is run by imsrunclient)
- So we "fake out" the EDG by having imsrunclient run a new program which conforms to the imsclient interface
- New program (imscop) actually just exec's imsclient for anything but an inventory search
- For inventory searches, instantiates the XMLClient class and has it handle things

Getting around ODL searches

- Normally, ModeSEARCH has already transformed the query (in the Query object) into ODL by the time imscop gets into the act
- A modification was added to ModeSEARCH to "freeze" a copy of the Query object itself
  - This is just a Perl object
  - It can be reloaded an used for whatever we want later
- We do the transformation into ODL anyway, because some searches don't go through ECHO
- The ECHO searches will just ignore it
- We didn't go query to ODL to XML
  - Too slow
  - Too hard
  - Data is already in Perl format in Query

xmlcop

- xmlcop only does the work of deciding whether or not the search should go to ECHO
- If it is supposed to, xmlcop instantiates the XMLClient class
  - It picks up the Query object and transforms it into AQL
  - It sends this off as a dataset search to ECHO
  - It parses the returned data and pulls out the list of datasets
  - It sends off a granule search for each dataset, transforming the XML from ECHO into client.obj output
  - Once these are all done, it writes the "finished" status to client.stat and exits

XMLClient classes

- AQLQueryDataset - can take a Query and generate AQL for a dataset search
- AQLQueryGranule - can take a Query and generate AQL for a granule search
- Inquisitor - encapsulates the ECHO SOAP interface and XML parsing
- Answer - Perl representation of the returned ECHO XML as a Perl object
- ClientResult - handles the transformation of the Answer object(s) into client.stat and client.obj

Generating AQL and XML

- There are lots of ways to transform data into XML
  - Most of them (e.g., generating SAX events) seemed complicated and unintuituve
- The ECHO AQL and XML are very stereotyped
  - Mostly just "plug this value into this field"
- So we just use Text::Template, extract values from the Query object, and expand the templates

The Inquisitor - creating XML

- So-called because it asks ECHO all the questions
- Also uses templates to handle the search XML
  - It gets the AQL coming in
  - So all it has to do is wrap a standard set of query XML around it
  - A very simple template that embeds the AQL inside the CDATA tags handles this just fine

The Inquisitor - Answers

- XML is fine for data transfer
- But we need Perl objects for faster access to the data
- We can parse the XML and create those objects
  - First for data sets
  - Then for the granules for each data set

The Inquisitor - XML parsing

- Uses an XML parsing technique that allows most of the specialized ECHO tags to be handled automatically, with the structure of the ECHO return mirrored into nested ResultGroup objects inside the Answer object
- THe EDG already can display these in the dataset attributes and granule attributes pages
- If the XML is bad, we just record this in the Answer and pass it on

ClientResult

- ClientResult collects up all the Answer objects and transforms these programmatically into a client.obj and client.stat file
- Each time a new Answer comes in, the client.stat and client.obj get rewritten, so we get partial results for free

From the user's point of view ...

- We see a regular CommStat, with ECHO as the DAAC
- When the search finishes, we see a dataset list (with most columns empty)
- If we go to dataset attributes, we see the attributes.
- If we ask to see the data granules, we get a granule list with granule name and dataset name
  - Each granule has its attrbutes as expected when we click "granule attributes"
  - The granules can't be ordered because we have no packaging data yet.

Where are we now?

- We can:
  - Build valids out of ECHO data, without relying on ECHO themselves to supply valids
  - Build a search as usual
  - Use the EDG's existing imsrunclient/CommStat code to run, monitor, and access the data output from a search by using imscop/XMLClient to do an end-run around the actual search mechanism
- We need to:
  - Design new code to handle the data arriving as nested result groups, so the dataset and granule lists can actually display the attributes
  - Address ordering
    - Can we get the package data into the EDG in a way that's compatible with the current ordering paradigm?
    - Or do we have to completely redesign the shopping cart code and the user interaction?